J2EE-Praktikum

EJB-Security

Peter Thiemann

Universität Freiburg

J2EE-Praktikum, WS2005/2006

Inhalt

EJB Security
Allgemeines

Sicherheitsziele

- Vertraulichkeit
- Integrität
- Verfügbarkeit

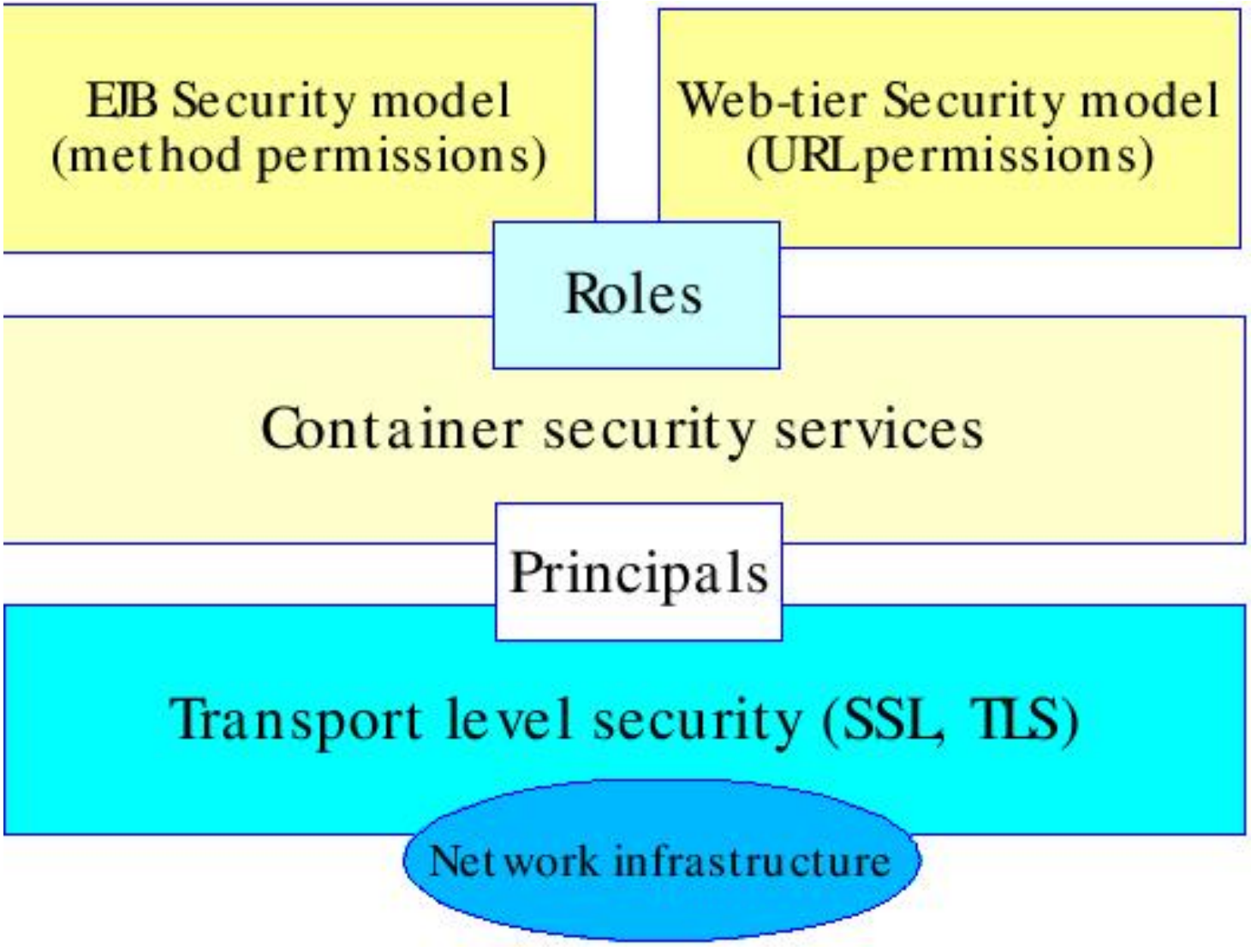
Sicherheitsmassnahmen

- Authentisierung / Feststellen der Identität
- Autorisierung / Feststellen der Zugriffsrechte
- Sichere Kommunikation

EJB Sicherheit
Ziele der EJB Sicherheitsarchitektur

- Lessen the burden of the application developer (i.e. the Bean Provider) for securing the application by allowing greater coverage from more qualified EJB roles. The EJB Container Provider provides the implementation of the security infrastructure; the Deployer and System Administrator define the security policies.
- Allow the security policies to be set by the Application Assembler or Deployer rather than being hard-coded by the Bean Provider at development time.
- Allow the enterprise bean applications to be portable across multiple EJB servers that use different security mechanisms.

EJB Sicherheit



EJB Sicherheit
Aufgaben der EJB Rollen

Bean Provider

Deklarative Spezifikation: Business-Methoden enthalten

- keine festen Zugriffsrollen
- keine Sicherheitslogik

Application Assembler

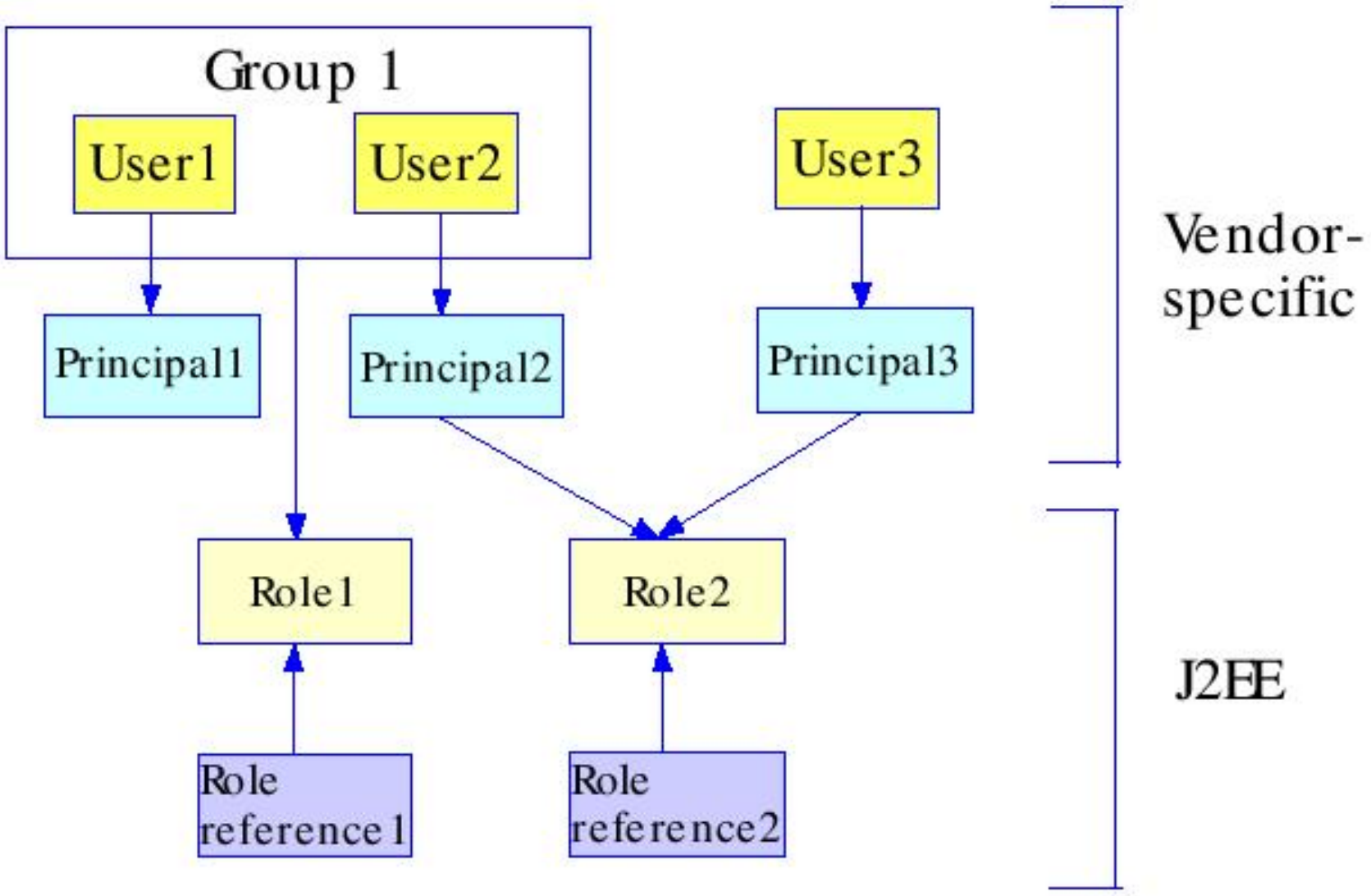
Legt durch Deployment-Descriptor fest:

- Sicherheitsrollen (Rolle $\mapsto$ Menge von Berechtigungen)
  Nicht verwechseln mit Rollen des Betriebssystems!
- Methodenberechtigungen (Erlaubnis, eine Gruppe von Methoden über eins der EJB-Interfaces aufzurufen)

Deployer

Legt Mitglieder (security principals) der Sicherheitsrollen fest

EJB Sicherheit



EJB Sicherheit

Rollenübergabe bei Methodenaufrufen

- Spezifiziert durch `security-identity` Element im DD
- Normalfall (`use-caller-identity`): Methode wird mit aktueller Rolle des Aufrufers ausgeführt
- `run-as`: Methode wird mit anderer Rolle (aus DD) ausgeführt
- Vgl. Unix set-user-id
- Exception, falls der aktuelle security principal keiner Rolle zugeordnet werden kann, der die Ausführung der Methode erlaubt ist
- Garantiert durch Container Provider ($+$ Managementwerkzeuge)

Schnittstelle für den Bean Provider

- Falls deklarative Sicherheit nicht ausreicht . . .
- Die API erlaubt keine Änderung des Security Principal, mit dem ein Bean-Methodenaufruf ausgeführt wird.
- Überprüfung des aktuellen Security Principal und Test der Rolle des Aufrufers ist (nur in den Geschäftsmethoden) möglich

```
public interface javax.ejb.EJBContext {
  java.security.Principal getCallerPrincipal();
  boolean isCallerInRole(String roleName);
  // ...
}
```

Schnittstelle für den Bean Provider

`java.security.Principal getCallerPrincipal()`

- `getCallerPrincipal` liefert den wirklichen Aufrufer, nicht das Ergebnis eines `run-as` Mappings
- aus dem Ergebnis kann mit `getName()` ein String z.B. für einen Datenbankzugriff ermittelt werden
- das genaue Ergebnis hängt von der Sicherheitsinfrastruktur ab

Schnittstelle für den Bean Provider

Beispiel für `getCallerPrincipal()`

```
public double getSalary(String employeeId) {
    java.security.Principal callerPrincipal =
                                ctx.getCallerPrincipal();
   String callerId = callerPrincipal.getName();
    // manager can read employee salary
    // employee can read only own salary
    if ( (ctx.isCallerInRole("manager")) ||
         ((ctx.isCallerInRole("employee")) &&
          (callerId == employeeId)) ) {
        // return salary information for the employee
        getSalaryInformationSomehow(employId);
    } else {
        throw new SecurityException("access denied");
    }
```

Schnittstelle für den Bean Provider

`boolean isCallerInRole(String roleName)`

- `isCallerInRole()` testet ebenfalls den wirklichen Aufrufer vro eventuellen `run-as` Mappings
- Verwendung für Tests,
  - deren Granularität feiner als eine Geschäftsmethode ist,
  - die außer der Rolle noch Nebenbedingungen, z.B. Ressourcenbeschränkungen beachten müssen
- Beispiel

```
public class PayrollBean ... {
  EntityContext ejbContext;
  public void updateEmployeeInfo(EmplInfo info) {
    oldInfo = ... read from database;
    // The salary field can be changed only by callers
    // who have the security role "payroll"
    if (info.salary != oldInfo.salary &&
        !ejbContext.isCallerInRole("payroll")) {
      throw new SecurityException(...);
    }
```

Schnittstelle für den Bean Provider
`isCallerInRole()`

- Potentielle Argumente von `isCallerInRole()` müssen dem Container angezeigt werden
- DD-Element `security-role-ref` (mit optionaler Beschreibung `description`)
- Gültig für ein Bean

Richtlinien für den Application Assembler

- Tätigkeitsbereich: DD
- Soll (muss aber nicht) definieren
  - *logische* Rollen (security roles)
    `security-role/role-name` (optional: `../description`)
  - Methodenberechtigungen
    Relation $\subseteq \mathcal{P}(R \times M)$ zwischen Rollen und Methodennamen eines Interfaces
- Gültigkeitsbereich: ganzes Deployment
- Definiert Verbindung zwischen `security-role-refs` und eigenen Rollen
- Deployer
  - kann Benutzer- und Gruppennamen des Systems auf Rollen abbilden oder
  - kann alles ignorieren bzw. umdefinieren.

Richtlinien für den Application Assembler
Methodenberechtigungen

- `method-permission` (Liste von Rollen oder `<unchecked/>`, Liste von Methodenspezifikationen)
- Methodenspezifikation besteht aus
  - EJBName
  - `*` (alle Methoden), Methodenname oder Methodenname mit Parameterspezifikation (bei überladener Methode)
- Beispiel

```
<method-permission>
  <role-name>employee</role-name>
  <method>
    <ejb-name>EmployeeService</ejb-name>
    <method-name>*</method-name>
  </method>
</method-permission>
```

Richtlinien für den Application Assembler

Methodenberechtigungen/2

```
<method-permission>
  <role-name>employee</role-name>
  <method>
    <ejb-name>AardvarkPayroll</ejb-name>
    <method-name>findByPrimaryKey</method-name>
  </method>
  <method>
    <ejb-name>AardvarkPayroll</ejb-name>
    <method-name>getEmployeeInfo</method-name>
  </method>
  <method>
    <ejb-name>AardvarkPayroll</ejb-name>
    <method-name>updateEmployeeInfo</method-name>
  </method>
</method-permission>
```

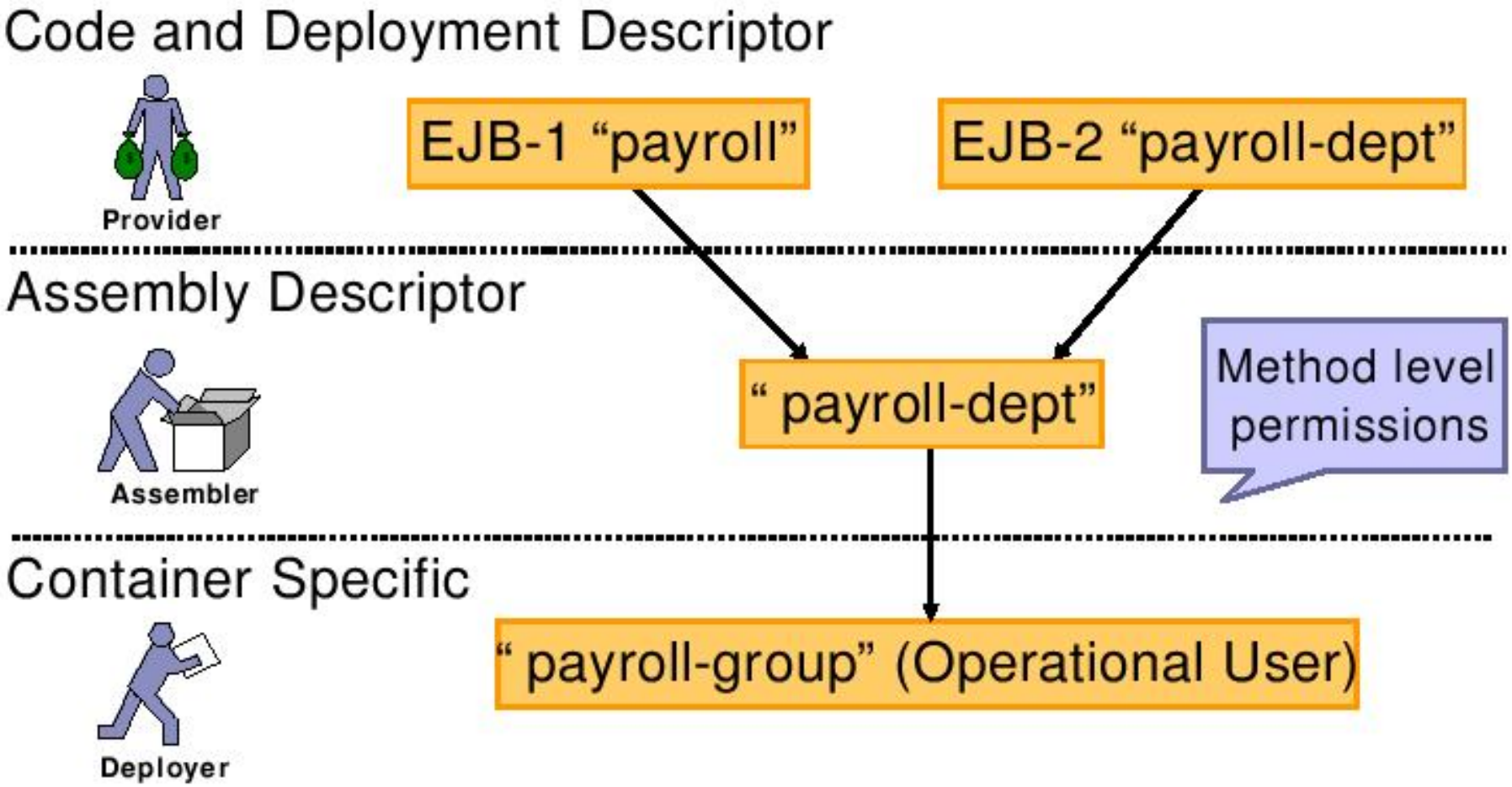
Richtlinien für den Application Assembler

Definition von `security-role-refs`

- Jedem Namen in einem `security-role-ref` muss eine Sicherheitsrolle zugeordnet werden
- Auch wenn Name und Sicherheitsrolle gleich sind
- Beispiel

```
<security-role-ref>
  <description>
     Members of this role have access to
     anyone's payroll record.
  </description>
  <role-name>payroll</role-name>
  <role-link>payroll-department</role-link>
</security-role-ref>
```

Richtlinien für den Application Assembler



Richtlinien für den Application Assembler

`run-as` versus `use-caller-identity`

- `run-as` gilt für ein ganzes Bean
- Der neue Principal wird **nicht** für die Methodenaufrufe innerhalb des Beans, sondern für Aufrufe zu anderen Beans verwendet.
- Beispiel

```
<security-identity>
  <run-as>
    <role-name>admin</role-name>
  </run-as>
</security-identity>
```

Richtlinien für den Deployer

- *The Deployer is responsible for ensuring that an assembled application is secure after it has been deployed in the target operational environment.*
- Auswerten der Sicherheitsarchitektur der Anwendung (DD) mit Deployment-Werkzeugen
- Abbildung auf Sicherheitsarchitektur der Zielmaschine
- Zuweisen einer Sicherheitszone
- Zuweisen von Personen/Personengruppen zu Sicherheitsrollen
- (Methodenberechtigungen bleiben gewöhnlich unverändert)
- Ggf. Füllen von Lücken, die der Application Assembler nicht spezifiziert hat

Richtlinien für den Container Provider

- Schutz der JVM und der Host-Maschine vor unerlaubtem Zugriff
- Unterstützung von SSL/TLS
  - zwischen Webbrowser und Webserverkomponenten (JSP-, Servlet-Container)
  - zwischen EJB-Containern
  - zwischen CORBA-Client und EJB-Container
- Container abstrahiert von der Verwaltung von Zugriffsrechten (LDAP, Datenbanken, . . .)
- Container ist für Authentisierung zuständig
  - Webkomponenten durch Name/Passwort
  - X.509
  - Kerberos

Zusammenfassung